Energy-Efficient Routing in GMPLS Network

Wang, Jiayuan; Fagertun, Anna Manolova; Ruepp, Sarah Renée; Dittmann, Lars

Published in:
Proceedings of OPNETWORK 2011

Publication date:
2011

Energy-efficient routing in GMPLS network

Jiayuan Wang, Anna Manolova, Sarah Ruepp, Lars Dittmann
Technical University of Denmark
E-mail: jwan@fotonik.dtu.dk

Abstract
In this paper, a GMPLS controlled core network model that takes energy efficiency into account is built using OPNET MODELER. By extending the standard GMPLS routing protocol -- OSPF-TE, we are able to spread desired energy related information over the local area, and subsequently use RSVP-TE for connection setup. The OSPF designing process of the simulation model is given in details, including necessary simplifications to enable a fast implementation while keeping the key characteristics of the GMPLS protocol behaviors. The efficiency of the proposed extensions are analyzed, and improved energy efficiency is shown through the results collected from OPNET MODELER simulation models.

Introduction
A Generalized Multiprotocol Label Switching (GMPLS) [1] controlled optical core network model is proposed, and built in this paper. The model utilizes Open Shortest Path First – Traffic Engineering (OSPF-TE) [2] as the routing protocol, and Resource Reservation Protocol – Traffic Engineering (RSVP-TE) as the signaling protocol. In order to combine energy efficiency [3] with the model built, the existing OSPF-TE protocols are extended to enable energy related information to be flooded over the local network area, and thus be used in the routing decision. The flooding procedure defined in OSPF-TE is revised and simplified to carry energy related information. Accordingly, the routing decision is based on "lowest energy consumption", rather than "shortest path", to achieve our goal of building an energy-efficient network.

In section II, the changes in the OSPF-TE protocol and routing decisions are introduced. The model built is discussed in details in section III, and results are shown in IV. Conclusions are given in section V.

OSPF-TE new extensions and routing calculation
OSPF-TE new extensions
In order to spread the energy related information using the OSPF-TE protocol, the protocol needs to be further extended to be able to carry the defined energy cost information.

In this paper, we consider opaque Link State Advertisements (LSAs) [5] of OSPF protocol for implementing the proposed GMPLS protocol extensions. New Type, Length, Value (TLVs) to TE extensions for OSPF [5] (TE LSA in this paper) are added. The TE LSA formats start with the standard LSA header, with TLVs as payload, as shown in Figure 1. The detailed explanations of each field can be found in [5].

TE LSAs defined two types of top-level TLVs. Our proposal is to add sub-TLVs to **Link** level TLV. Thus, the energy information is carried by setting up new sub-TLVs inside Link TLV (type 2) of TE LSAs [5]. *Type* in sub-TLVs is considered in the range of 32768-32777, as defined for experimental use. *Length* could be defined as 4 octets, as most of the other sub-TLVs defined in this level. *Value* is defined to carry the energy information, normally a value assigned to represent energy "cost".

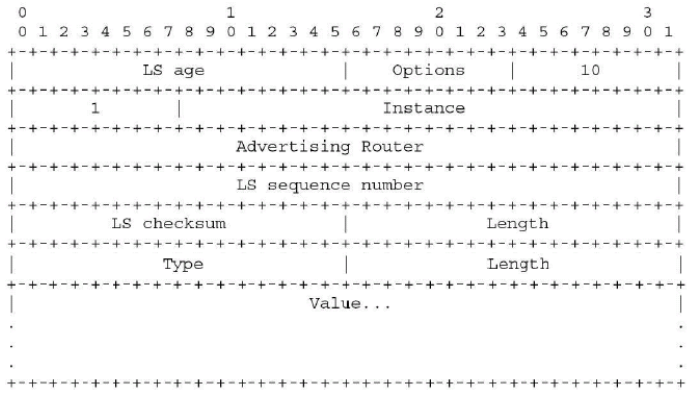

Figure 1: TE LSA format [3]

The flooding procedure follows the standard procedure of OSPFv2 flooding. In the simulation model, no designated routers are selected. Thus, LSAs are flooded in broadcasting manner. *MinLSInterval* defines the minimum time to update the same entry in the Link State Database of each node, so that duplicated LSAs received from multiple neighbors can be ignored. After receiving a new LSA, the node decides whether to forward the LSA or discard it according to the LSA type and the carried timestamp. Link State Database is updated upon receiving valid LSAs. Please refer to [2], Section 13 for details. According to the definition of sub-TLVs for TE LSA, the defined sub-TLVs may occur at most once. TE LSAs are flooded upon changes of contents. The triggers for originating new TE LSAs are implemented with fixed time-outs. For nodes that do not support proposed TLVs, the TLVs will be ignored.

Routing calculation
Routing calculation is based on standard Dijkstra algorithm used in OSPF-TE protocol. The "cost" assigned in route calculations will be updated by the energy cost carried in TE LSAs. The proposed model defines two types of cost value. The number could represent green energy source (0), and dirty energy source (1). Or simply cheaper energy source (0) and more expensive energy source (1), which applies to both link and node cost. In the proposed model, both link energy cost and node energy cost should influence the routing decision. However, as Dijkstra algorithm only takes one single value per

edge, the weight assigned to Djikstra algorithm should be revised to fit the need. Thus, each edge cost is represented by *link cost + 0.5 * source node cost + 0.5 * destination node cost*. The whole route cost will be a sum of all the edges cost. In this way, both node energy cost and link energy cost are taken into consideration.

An OPNET MODELER GMPLS model

The network simulation model is developed in three different layers:

Network model;
Node model;
Process model;

Network Model

The simulated environment is shown in Figure 2, which consists of twenty-eight nodes spanning over Europe. Each node is configured the same, with ability to support up to seven different neighbors/links.

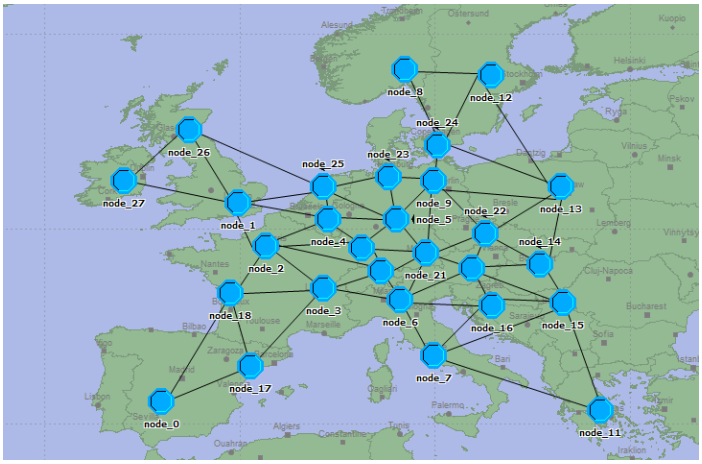

Figure 2: Network model [COST 266 topology]

Node Model

As shown in Figure 3, node model consists of a request generator (ReqGen), a routing module (Routing_module), an OSPF entity, a RSVP-TE entity, seven receivers and seven transmitters.

The request generator is responsible for generating connection requests. OSPF entity creates and processes the proposed TE LSA packets. It sends updates of routing table to the routing module, and ensures all the TE LSAs are flooded all over the network. The routing module initializes and calculates the routing table maintained in each node. Once the connection request arrived, it is up to the RSVP-TE entity to calculate the route, set up the connection, allocate wavelengths and reserve the resources. The routing decision is based on the routing table updated by the OSPF module. The transmitters and receivers support up to seven direct neighbors/links.

Process Model

Request generator

Request generator process schedules self-interrupts to generate traffic using an exponential distribution, and sends remote interrupts to invoke the RSVP-TE process once a request is generated. The Finite State Machine (FSM) can be seen in Figure 4. After checking the node status and initialize variables in *Init* state, the state enters *Idle* state, where it schedules self-interrupt. Request is generated in *Gen_req* state, where the remote interrupts to RSVP-TE model is also sent.

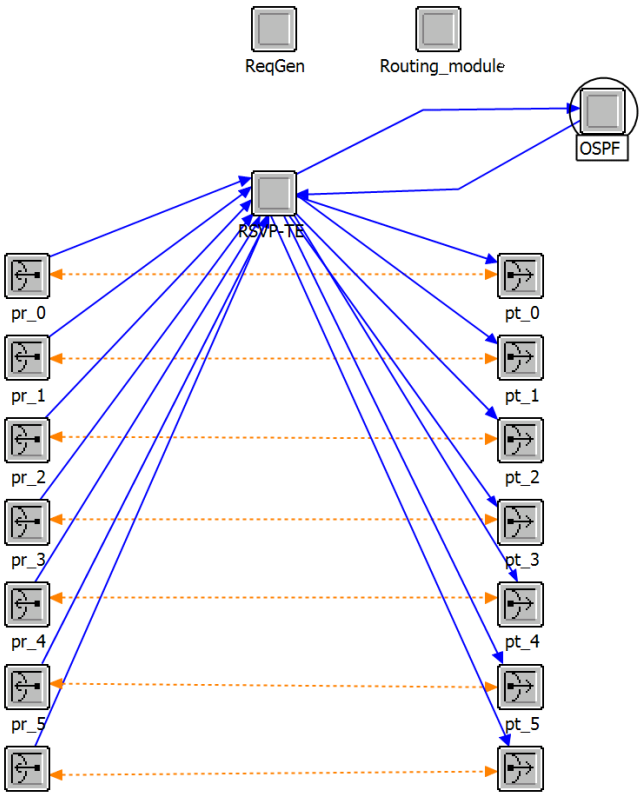

Figure 3: Node model

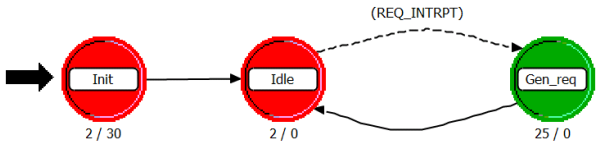

Figure 4: ReqGen process model

Routing module

The routing module is the key entity in making routing decisions. The module maintains a graph to be used for the Dijkstra algorithm built in OPNET MODELER. The graph creates a mapping between Dijkstra edge-vertex model with the topology implemented. After the routing graph is

initialized, the routing module sends a remote interrupt to invoke all the other modules. The FSM for the routing module is shown in Figure 5. In *init* state, variables are initialized and a routing table is built, describing the relationship between the simulated network and Dijkstra routing calculation. The state also checks the route's status, if the router is enabled in the network, the state goes to *operational*, and otherwise the state goes to *End_Sim*. In *Disable Edge* state, the links local resources are checked. When all the wavelengths are used up, the edge value in Dijkstra graph will be disabled.

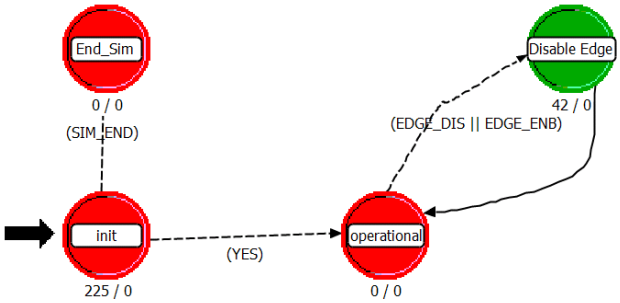

Figure 5: Routing module process model

OSPF module

The OSPF module processes all the OSPF messages flooded over the network. New TE LSAs are created in this node, and LSA database are maintained here as well. Upon flooding the TE LSAs over the whole network, new energy cost is updated in each LSA database, and thus updated in Dijkstra edge-vertex graph, which is also maintained in routing module.

OSPF version 2 is implemented in the OSPF module. For simplification, "Hello" process is skipped, and the process starts with automatic detection of links and neighbors (which is implemented in Routing module). The OSPF module only needs to look up the list in Routing module to achieve "Hello" process results. "Database Exchange" process is also skipped. The neighbors' routing table is exchanged directly by flooding process. The main function implemented is the flooding process. According to the information needed in the routing calculation, two types of energy related LSAs are flooded, node energy cost LSAs and link energy cost LSAs. Node energy cost LSAs are only flooded to their direct neighbors, since the node energy cost information is only needed when calculating the energy edge cost. Upon calculating energy edge cost, the information is flooded by link energy cost LSA over the whole network area. The flooding procedure follows standard procedure defined in [5]. Both of the extended TE LSAs are flooded at a fixed time interval, which can be user defined before running the simulation. Other parameters that can be user defined are, *MinLSInterval* [3] time, retransmission time, energy cost value.

The FSM of OSPF module is shown in Figure 6. In *Init* state, all variables are initialized. Once a remote interrupt from Routing module is received (indicating the finishing of initialization of routing module), the state enters *Begin*, where the list of neighbors and links are initialized, together with the Dijkstra graph. The self-interrupt for the power source changes are also scheduled. The flooding process is accomplished by all the green states surrounding state *Exchange.* State *Create* is responsible for creating new LSAs, lists for keeping track of LSAs (Link Sate Database) and the corresponding acknowledgements (ACKs) are also built. In a fixed power change time, new node LSAs and link LSAs are created. The link LSAs are created based on a master-slave relation. The link power type (both directions) can only be changed by either the source or destination node, which has a bigger node ID value. State *WaitOver* is reserved for standard OSPF Database Exchange process, which will not be used in this module. All the received packets are processed according to their types. After the state *CheckType*, which returns the type of the received packet, the LSA packets will be processed in state *LS_update*, and the ACKs will be processed in state *LS_ack*. The state *LS_update* includes judgments of the received LSA, whether it is valid (new or updated) or should be discarded. All the valid LSAs are stored in Link State Database and re-flooded to neighbors, together with updates of Dijkstra graph. Once a new LSA is flooded out from the router, the ACK status is recorded in the neighbor ACK list, and the possible retransmissions are prepared. In state *LS_ack*, the router checks each received ACK from neighbors. If the ACK is not duplicated, it is registered in the neighbor ACK list. Once ACKs from all the neighbors for a particular LSA are received, the entries for neighbor ACK list and retransmissions are deleted. In state *Retransmission*, retransmissions are sent upon receiving retransmission timeout. The packet for retransmission is chosen by looking up the neighbor ACK list.

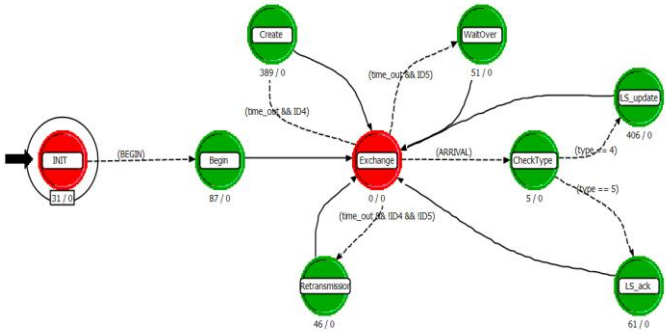

Figure 6: OSPF module process model

RSVP-TE module

The RSVP-TE module is responsible for processing connection requests. It responds to either *PATH* or *RESV* messages to allocate labels and reserve resources. Each

request is handled by child processes. The RSVP-TE module also calculates routing decisions based on the value updated from OSPF module. The cost calculation results are collected and recorded. In comparison, a parallel process model that uses the shortest path algorithm is also built. The corresponding energy cost based on the result of shortest path calculation is also recorded.

The FSM is shown in Figure 7. In state *Init*, the graph for mapping between the links in the simulated model and the edge values in Dijkstra graph is built. The same as in OSPF module, the state enters *Idle* state after receiving a remote interrupt from Routing module. The connection requests are handled in the state *Req_handle*, including processing of *PATH* and *RESV* messages. If stream interrupt is received, the state passes through state *check*, and process packets from OSPF module in state *OSPF_module*, and packets received from other sources in *Intrpt_steer*. Once the connection is expired, the releasing of resources is performed in the state *Rem_conn*.

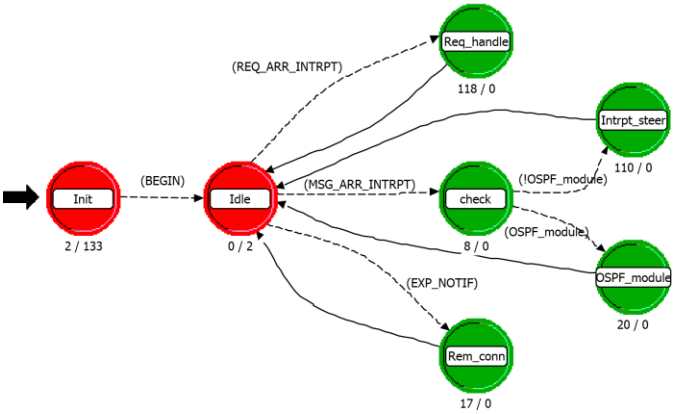

Figure 7: RSVP-TE process model

Simulation Results

In this section, the results of our proposed model are discussed and evaluated, with focus on the energy cost reduction. As a comparison to the proposed energy efficient routing algorithm, the results using hop based Shortest Path (SP) algorithm are also collected. Both of the scenarios are compared in the performance of energy cost and routing hops.

In the twenty-eight nodes model, the request inter-arrival time for each node is configured to be exponentially distributed with a mean value of 10s, and duration of each request is exponentially distributed with a mean value of 10s. The wavelength per link is configured to be 10. Each LSP request destination is uniform distributed between all possible destinations.

Under scenario 1, our proposed energy efficient algorithm is used. We set up OSPF to flood energy cost information at a fixed interval of 10 minutes, and the energy cost is selected from a random value of either 0 or 1. Node energy cost is flooded to its direct neighbors, and link energy cost is flooded over the whole network. The cost taken into Dijkstra calculation is a combination of both node and link energy cost, as stated in the section routing calculation. *MinLSInterval* is defined to be 30 seconds.

Under scenario 2, the same settings as above are used. Instead of calculating the route based on energy cost, the route is calculated based on the least hops. With different routes selected compared to scenario 1, the energy cost values are looked up and added along the routes as in scenario 1. Thus, the energy cost result using the SP algorithm is obtained. Both of the results of energy cost are shown in Figure 8, with a simulation runtime of 2 hours.

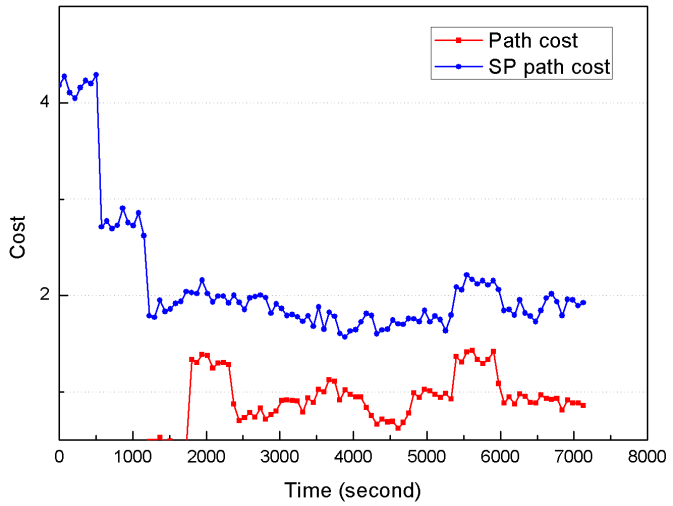

Figure 8: Energy cost

As shown in Figure 8, due to the initialization of different modules, the results line can be unstable at the beginning, as we see in the figure. The results become stable from 2000 seconds, and the energy cost using the proposed energy efficiency algorithm is indeed lower than that using the SP algorithm. If we consider the energy cost 0 to be green energy source, and energy cost 1 to be dirty energy source, the result shows the proposed algorithm can effectively route the traffic away from dirty energy sources, since the average energy cost is less than 1.

However, with the cost of energy being lowered using the proposed algorithm, the length of routes selected has increased. As shown in Figure 9, the average number of hops of each selected route is higher by using the proposed algorithm than that using the SP algorithm. The hop count number of using SP algorithm is higher than the one using proposed algorithm at the beginning, which might due to the instability of the system. With higher hop count number using the proposed algorithm, the average hop count can still be

maintained below 6, which is one or two hops longer than the results using SP algorithm, but are still below the network dimension, which is 8 hops.

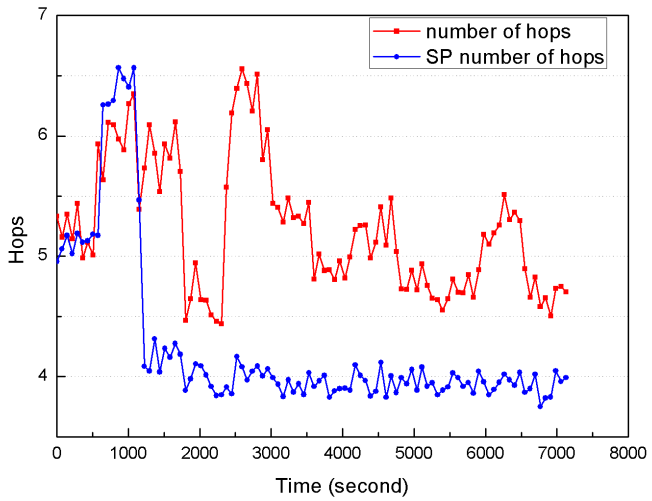

Figure 9: Routing hops

Conclusion

In this paper, a GMPLS controlled network simulation model is built with OPNET MODELER. In order to enable energy efficiency in the built network, an energy efficient GMPLS protocol extension is proposed and simulated. Compared to the traditional SP algorithm used in routing decisions, the results show our proposed algorithm can effectively lower the energy cost by either routing the traffic away from dirty energy sources, or by choosing lower energy cost routes (depending on the definition of assigned energy cost value), while still maintaining acceptable routing hops numbers.

The obtained results are based on a random energy cost assignment. In order to get results closer to the real world implementation, more complex energy cost models could be employed as input to run the simulations, as part of our on-going work. Optimization could also be done in order to further reduce the hop count.

References

[1] E. Mannie, “Generalized Multi-Protocol Label Switching (GMPLS) Architecture”, RFC3945, Oct. 2004.
[2] J. Moy, “OSPF Version 2,” RFC2178, Apr. 1998.
[3] S. László, “Energy-efficient Networking: An Overview”, Acta Universitatis Sapientiae, Oct. 2010.
[4] R. Coltun, “The OSPF Opaque LSA Option,” RFC2328, Jul. 1998.
[5] D. Katz, “Traffic Engineering (TE) Extensions to OSPF Version 2,” RFC2370, Sep. 2003.